**write at least one paragraph about the differences between the code in ec2train1.py and the code you used in Step 1. You may work on projects that require adaptation between EC2 and SageMaker code, and understanding the differences in this project can help you prepare for that kind of adaptation.**

I choose t2.micro instance because it's a free tier

The differences between the two codes:

1. calculate the loss and accuracy by get the floar devision instead of normal devision  
   total\_loss = running\_loss // len(test\_loader)  
   total\_acc = running\_corrects.double() // len(test\_loader)
2. pass the info to the logger :   
   logger.info(f"Testing Loss: {total\_loss}")  
   logger.info(f"Testing Accuracy: {total\_acc}")
3. instead of (log = Report(epochs))  
   using the root = train\_data\_path  
   train\_data = torchvision.datasets.ImageFolder(root= train\_data\_path)  
   transform=train\_transform)
4. instead of using root = test\_data\_path  
   train\_data = torchvision.datasets.ImageFolder(root=test\_data\_path, transform=train\_transform)
5. using direct variables  
   train\_loader, test\_loader, validation\_loader=create\_data\_loaders('dogImages',batch\_size)  
   instead of using args data  
   train\_loader, test\_loader, validation\_loader=create\_data\_loaders(args.data, args.batch\_size)

**Write at least 1 paragraph describing how this function is written and how it works.**

* Lambda is a very important AWS service that allows external users and applications to interact with the ML models in your project.
* Lambda functions can act as an intermediary between users and ML models. They can take inputs from users and pass those inputs to your deployed endpoints. Then, they get outputs from your deployed endpoints and pass them on to users.
* Like any computing resource, Lambda functions can get overwhelmed if there’s too much traffic. If they’re overloaded with traffic, their latency, or response time, will increase. Later, we’ll talk about strategies for dealing with high traffic and achieving low latency.

To deploy a Lambda function:

* You can search for Lambda in the AWS search bar. After you click on Lambda, you'll see the AWS Lambda dashboard
* In this dashboard, you should click "Create function" to create a new Lambda function. You will see a screen that allows you to make decisions about the configuration of your new Lambda function.
* You can select "Author from scratch", as well as any function name, and a recent Python runtime like Python 3.8 or Python 3.9. Then you can click "Create function" again to create the function. After creating the function, you'll see a screen that allows you to adjust the code, deploy changes, and test your function.
* You can adjust the code in the "lambda\_function.py" tab to change what the Lambda function does. After you adjust the code, you can click "Deploy" to save the changes.

**Write about the security of your AWS workspace in your writeup. Are there any security vulnerabilities that need to be addressed? Think about some common security vulnerabilities.**

Security roles identities that have specific permissions in your AWS account.

* When you create a notebook in SageMaker, you’re doing it with a particular role. If you own your AWS account completely, then you should be doing this with the “root user” role. Root users have universal permissions to adjust anything in the account.
* If you create a Lambda function that invokes one of your endpoints, that Lambda function will have a specific role associated with it. If that role doesn’t have permission to invoke your endpoint, then you’ll get an error.
* If you want to adjust the permissions associated with a particular role, you need to work with IAM “policies”. A policy is a set of permissions to perform specific actions. If there’s a role that’s not able to perform an action that it needs to perform, you need to “attach” a security policy to that role.

Top 7 AWS security vulnerabilities based on real-world tests:

* Sensitive data in X in plaintext
* Publicly accessible SQS
* Publicly accessible S3
* Privilege escalation
* Wide cross-account access
* Lack of monitoring
* Lack of Transfer Lock

**When you set up concurrency and auto-scaling, you will make several choices about configuration. Write about the choices you made in the setup of concurrency and auto-scaling, and why you made each of those choices.**

Adding Concurrency to Lambda Functions:

1. Start by opening Lambda and clicking "Functions" in your AWS Workspace.
2. Open a function - either an existing one or a new one that you create.
3. In the Configuration section, select Concurrency.
4. Click the Versions tab to create a new version of your Lambda function.
5. Select the Concurrency tab again, and add reserved or provisioned concurrency to your function.
6. Test your Lambda function and make sure that it's still functioning properly.

I choose 5 for your reserved concurrency, and 2 for your provisioned concurrency. Remember that your provisioned concurrency always needs to be less than your reserved concurrency.

Choosing a high number for provisioned concurrency is suitable for very high-traffic projects, because it will turn on instances that can be used by your Lambda function any time. However, provisioned concurrency is expensive, so you might want to keep this number low.

Choosing a high number for reserved concurrency incurs no additional cost. It will allow you to choose a high number for provisioned concurrency, since provisioned concurrency must be lower than reserved concurrency.

Choosing lower numbers for either or both types of concurrency would be more suitable for lower-traffic projects, or projects with lower budgets.

Autoscaling:

Next, navigate to SageMaker and open Inference > Endpoints to find the endpoint you just deployed. Click on the endpoint, and you should see some tables with settings and metrics related to your endpoint. Find the table labeled Endpoint runtime settings

I choose 1-3 min-max instances

Select your deployed endpoint, and select Configure auto-scaling.

This will take you to a dashboard where you can configure auto-scaling for your endpoint.

You can use any configuration that you think will be appropriate for your endpoint.

You might want to use 30 for the scale-in and scale-out cool down periods, and 10 for the other parameter (target invocations per instance).